Digit Recognition Using Single Layer Neural Network with Principal Component Analysis

Vineet Singh, and Sunil Pranit Lal
School of Computing Science, Information System and Mathematics,
The University of the South Pacific,
Laucala Bay, Suva, Fiji
{vineet.singh, sunil.lal}@usp.ac.fj

***Abstract* – This paper presents an approach to digit recognition using single layer neural network classifier with Principal Component Analysis (PCA). The handwritten digit recognition is an important area of research as there are so many applications which are using handwritten recognition and it can also be applied to new application. There are many algorithms applied to this computer vision problem and many more algorithms are continuously developed on this to make the handwritten recognition classify digits more accurately with less computation involved. The proposed model in this paper aims to reduce the features to reduce computation requirements and successfully classify the digit into 10 categories (0 to 9). The system designed consists of backward propagation (BP) neural network and is trained and tested on the MNIST dataset of handwritten digit. The proposed system was able to obtain 98.39% accuracy on the MNIST 10,000 test dataset. The Principal Component Analysis (PCA) is used for feature extraction to curtail the computational and training time and at the same time produce high accuracy. It was clearly observed that the training time is reduced by up to 80% depending on the number of principal component selected. We will consider not only the accuracy, but also the training time, recognition time and memory requirements for entire process. Further, we identified the digits which were misclassified by the algorithm. Finally, we generate our own test dataset and predict the labels using this system.**

Keywords – Neural Network, PCA, Digit Recognition

I. INTRODUCTION

Handwriting recognition has become one of the most interesting directions in solving computer vision problem in the field of image processing and pattern recognition. This technique is used in many potential applications such as bank cheque analysis, US post mail sorting [12] and handwritten form processing [2]. There are many approaches has been applied to this with high accuracy [1, 2, 3, 4, 5, 6, 7], however there are rooms for enhancement. We got the handwritten recognition idea from Kaggle competition (https://www.kaggle.com). Kaggle is a competition where we can take part and see where our algorithm stands compared to other researchers.

In [3], the proposed system uses multiple feature extraction techniques and multiple layer perception (MLP) neural networks achieved a good accuracy rate. The feature extraction methods used were multi zoning modifies edge [4], structure characteristics [7], image projection [5], concavities measurements [8] and MAT-based gradient directional features [5].

LeCun and his team, in [9] compare several classifiers applied on handwritten digit recognition, from which boosted LeNet 4 gave the best accuracy of 99.3%. Boosted LeNet 4 model combined multiple LeNet classifiers which had multiple convolution layers neural network.

Different classifiers and combination methods were used with PCA to reduce features for faster training time. Our goal is to use computationally less expensive neural network and PCA with minimum dimension for digit recognition to improve the accuracy optioned from [1]. The accuracy for this system was 91.20% using K=64 input features, 35 hidden layers nodes and 4 output neurons.

The MNIST dataset of handwritten digit with labels 0 to 9 was used for training and testing. The dataset has 60000 training set and 10000 test set images. Each image is of size 28 x 28 pixel grayscale image (0 – 255). There is a general problem in prediction of similar digits such as 5 and 9, 1 and 7 and others. The handwriting of individual personnel can also influence the prediction as a digit can be written in different ways, such as digit '5' is written as '**5**', '**5**', '**5**' or '5'.

In this paper we used single layer neural network classifier with PCA as shown in Fig. 1 to extract the features and train using back-propagation algorithm.

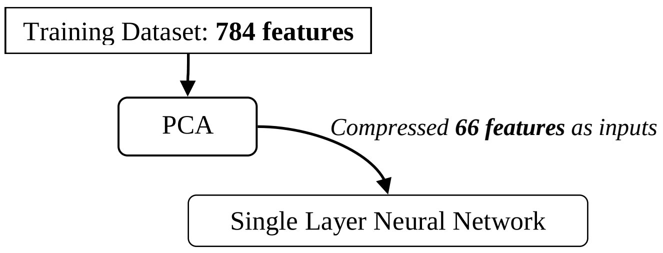

Training Dataset: **784 features**
PCA
*Compressed **66 features** as inputs*
Single Layer Neural Network

Fig. 1. Overview of proposed model

This paper is organized as follow: section II briefly discusses the data processing techniques used. The proposed classifier with PCA is demonstrated in section III. Section IV shows the training of the classifier and section V explains how the experiments were carried out. Followed by results, discussion and concluded with future directions.

II. Data Preprocessing

The digit may be written in different ways, therefore the data needs to be normalized to eliminate noise in the data and get all the dataset in a fixed format. A portion of training dataset is shown in Fig. 2.

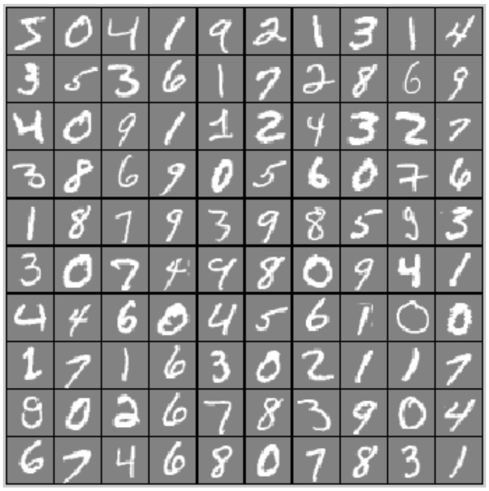


Fig.2. Sample of normalized training data loaded in MATLAB

A. Normalization of MNIST Dataset

The MNIST dataset has 60000 training samples and 10000 testing examples which was used for training and testing the model. Each sample is normalized and centered in 28 x 28 pixel grayscale image resulting in a total of 784pixel per image and each pixel value ranged from 0 to 255. After normalization, each pixel value range from 0 to 1.

B. Creating Test Samples using Paint Application

In addition to MNIST dataset, I have created my own test dataset which includes twenty test samples using the paint application. Different digits were written on 28 x 28 pixel image using a black color with a white as the background. This was to test if the trained algorithm could predict actual handwritten digit on paint application which required preprocessing of the digit image.

1) Grayscaling

The images were loaded in MATLAB and 32-bit color images were transformed into grayscale image of 28 x 28 pixel similar to the training set format. The images were further normalized by simply dividing each pixel by 255 resulted in pixel between 0 to 1. The transformation of image from paint application to grayscale image in MATLAB is shown in Fig. 3.

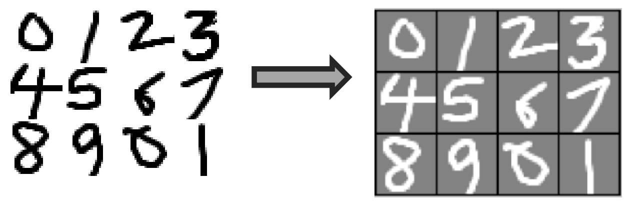


Fig. 3. Original Paint data to normalized grayscale data in MATLAB

III. Proposed Classifier

The single-layer neural network classifier in Fig. 4 has been implemented as neural network with three layers (one input, one hidden and one output layer). The resulting output from PCA shown in Fig. 1 is the input of the neural network which is 66 input neurons. There are 99 nodes in hidden layer. We chose the neurons based on the experiments with different hidden nodes and selecting the nodes which gave the highest cross validation (cv) accuracy. Forward propagation is used to classify the digit with respect to the output layer neurons. The output layer consist ten nodes, each corresponding to ten digits (0 to 9). The ten neuron's output is calculated and classified digit corresponds to neuron with highest output value (highest probability).

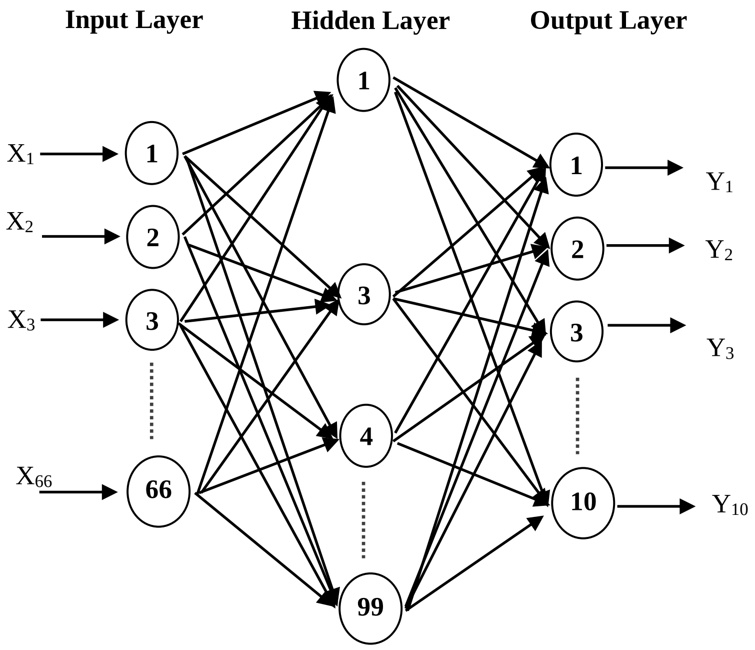


Fig. 4. Proposed single layer network

A. Feature Extraction Using Principal Component Analysis

Principal component analysis (PCA) is fundamental multivariate data analysis method which is used in various area in neural network and machine learning. It is used to reduce the dimensionality of the existing dataset. PCA can be applied to the digit images by projecting the item onto smaller dimension.

1) PCA Algorithm

The PCA algorithm can be implemented in the following steps [13]:

i. Calculate the mean for each dimension and subtract each training sample with the mean as shown in equations 1 and 2 respectively.

$$u[m] = \frac{1}{N} \sum_{i=1}^{N} X_i \qquad (1)$$

$$X = X - u[m] \qquad (2)$$

ii. Find the covariance matrix and get the eigenvector (V) and eigenvalue (D) as in equation 3 and 4.

$$C = \frac{1}{N} X * X^T \tag{3}$$

$$V^{-1} * CV = D \tag{4}$$

iii. Sort the eigenvector and eigenvalue and select the Kth most significant eigenvectors. Project data X into K dimensional by multiplying X with top K eigenvectors.

$$Z = X * V(1:K) \tag{5}$$

where *u[m]* is the mean of training set, *X* is the sample of training set, *N* is number of sample acquired, *C* is the covariance matrix, *V* is the eigenvector of *C* and *D* is the eigenvalue of *C*, *K* is the value of principal component and *Z* is the eigenvector of *X*.

IV. Training the Classifier Using Back Propagation Algorithm

Artificial Neural Network composed simple neurons connected to each other with its own connection strength whose function is determined by network structure. This is used for different problems such as in health application for analyzing the heart disease in pattern recognition. [11] We trained the neural network using back propagation algorithm where the learning takes place by the adjustments of randomly initialized weights such that the classifier error is minimized. In back-propagation neural network, the learning takes place in two parts. First, a training sample is presented to the input layer. The network propagates from layer to layer until output pattern is obtained in output layer. If the actual output and desired pattern is different, then the error is calculated. The error is propagated backwards and weights are modified as the error is propagated.

A. Learning using Back Propagation

We initialized the weights randomly between -0.5 to 0.5. The uni-polar sigmoid activation function shown in equation 6 is selected comparing the accuracy rate of different function in [10]. The actual output of the neurons in hidden layer and output layer is calculated by activation function using forward propagation. The error gradient is calculated using the actual output and the desired output. The error is propagated backwards in the network simultaneously calculating weight correction. Finally, all the weights are updated and this is repeated for each training sample in all epochs. The error gradient is minimized in each iteration using fmincg function of MATLAB.

$$g(x) = \frac{1}{1+e^{-Z}} \tag{6}$$

$$Z = \sum_{i=1}^{n} X_i W_i - \theta \tag{7}$$

where Z is the input value in activation function, n is the number of neurons, X_i is value i^{th} neuron, W_i is i^{th} weight isosiated to the neuron is and θ is threshold applied to neuron.

V. Experimentation

A. MNIST Dataset

The dataset originally consist of 784 features measured over 60000 training set and 10000 test set. We compressed the number of features to 66 features using PCA which was used as inputs for the neural network. Since the output has 10 nodes, the dataset has labels from 0 to 9. The distribution of training sample of 10 digits is shown in table I number of each digit was there in the dataset.

TABLE I
NUMBER OF EACH DIGIT IN THE DATASET

Digit	***No. Training Sample***	***No. Test Sample***
0	5923	980
1	6742	1135
2	5958	1032
3	6131	1010
4	5842	982
5	5421	892
6	5918	958
7	6265	1028
8	5851	974
9	5949	1009
Total	60000	10000

B. Tools Used For Implementation

MATLAB was extensively used for coding due to its advance libraries of the mathematic functions. Due to large dataset, high memory was required. Each pixel requires 20bytes so for training set of 60,000 x 784 is equal to 942MB of memory is required for training set and a total 157MB for test set, resulting more than 1GB memory required to load the dataset on MATLAB followed by all other training and testing process.

C. 10-Fold Cross-Validation

Validation techniques are important phase in training fundamental problems in pattern recognition for model selection and performance estimation. This is used to prevent overfitting or underfitting of the model. We choose 10-fold cross-validation method used for training and testing our different model and picking the model with lowest average error. The advantage of K-Fold cross validation is that all the samples in the dataset are eventually used for both training and testing.

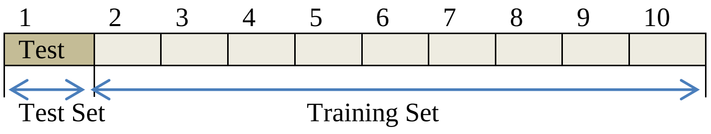


Fig. 5 10-fold dataset (9 for training and 1 for testing)

We divided the 60, 000 training dataset into 10 subsets, each set has 6,000 examples as shown in Fig. 5. For each cross-validation experiment we used one subset as the test set and remaining as training set for all 10 different folds as captured in Table II.

1) *Average Error Rate*

After calculating all errors from the model in 10-fold cross-validation for 1 experiment, the average error rate was calculated using the formula (equation 8) where ***K*** is number of fold, 10 in this case and $\boldsymbol{E_i}$ is error occurred in each of the i^{th} fold testing.

$$E = \frac{1}{K}\sum_{i=1}^{K} E_i \qquad (8)$$

D. *Principal Component Analysis*

PCA was used to reduce dimension from 784 to lower value ease computation. After executing PCA on the dataset, we found that 281 features retained over 99% of variance, 103 features retained 95% of the variance and 53 features retained 90% of the variance. The Fig. 6 shows the percentage of variance retained by different numbers of principal components.

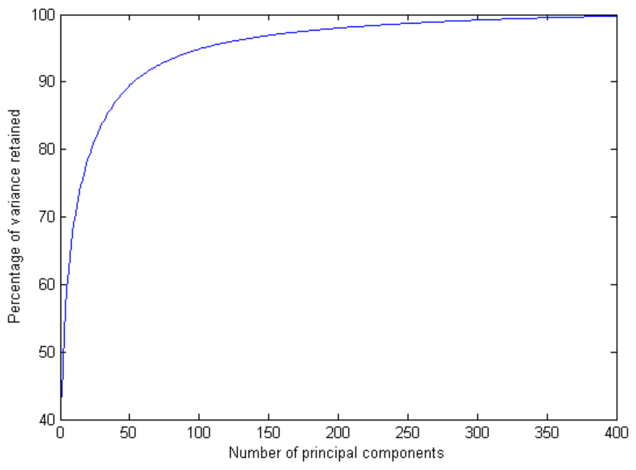


Fig. 6. Variance retained after PCA on training data

E. *Training and Testing phase*

There were multiple training sets to find best different parameters. The training was both done with and without PCA to compare the difference in the time taken with respect to accuracy rate. The general procedure taken for training in this paper is captured below. The training and testing procedure used in this paper is captured below.

1) *Training Phase*

The training of the proposed algorithm was done on 60,000 MNIST dataset following the steps in table II.

TABLE II
TRAINING MNIST DATASET SETS

1. For each experiment with different parameters.
2. For each K^th-fold cross validation (*see Table II*)
3. We initialize the random weights and set other parameters of the classifier which are hidden neurons (*h*), iteration (*i*) and number of inputs (*K*) from PCA
4. A sample, X^i from 54,000 cross-validation (cv) training sample is passed through the classifier
5. The classifier gives result of output of that sample using the activation function.
6. If the output is different from the desired output, then the error gradient is calculated.
7. The error is propagated backwards in the network update the weights with respect to the error.
8. Steps 4 to 7 is repeated for all 54,000 cv training samples and the weights are updated according the error.
9. The training runs for different epochs based on the iteration *i* for this experiment, minimizing the classification error.
10. After all epochs, the forward propagation is taken to classify the 6,000 test sample
11. Calculated the accuracy and error rate in classifying in step 10.
12. Repeated step 2 to 11 for all 10 folds, where one K^th subset of 6,000 samples is test data and remaining 54,000 samples are training data.
13. The average cv error of 10-folds is calculated and all parameters of this model were saved.
14. Steps 1 to 13 are carried out for different experiments with different *h, i and K values*, such that we get maximum accuracy with minimum computation.
15. After all experiments, results of the models were compared and model with best result was chosen.

The first set of training was done with model without using PCA for different number of epochs and the number for neurons in the hidden layer was be obtained from the rule:

$$N = \frac{m+n}{2} \qquad (9)$$

Where *m* and *n* are the number of neurons in the input and output layers respectively.

2) *Testing Phase*

The testing of the proposed algorithm was done as follows:

TABLE III
TESTING PROCEDURE

1. The model with best accuracy rate was chosen for testing in 3 set of different test samples. The MNIST 10,000 test set, 30 paint digit images and in Kaggle competition with 28,000 test data with unknown labels (https://www.kaggle.com).
2. The model was used to predict the digit 0-9 using the test set from three different source as in 1.
3. The predicted digit is compared with the actual digit to get the classification accuracy for all the test samples in MNIST and paint dataset. For Kaggle test set, the predicted 28,000 digits are uploaded on the on Kaggle and the accuracy result is given back.

F. Training Without PCA

The first experiment was without PCA which followed the same procedure as discussed in training phase, only the input nodes having all 784 pixel inputs. The single layer neural network model has 784 input nodes, 397 hidden nodes and 10 output nodes, with 0.1 learning rate and 2 as regularization lambda value. The following table IV shows the accuracy and time taken for different iteration.

TABLE IV
RESULT OF MODEL WITHOUT PCA

Iteration	Average CV Accuracy (%)	Time Taken (second)
200	97.433	9632.46449
400	98.003	18544.6042
600	98.105	27256.2739
800	98.113	35923.0287
1000	98.274	45011.5432

G. Training Neural Network With PCA

The second experiment was using neural network with Principal Component Analysis (PCA). The selection of number of principal component (K) which becomes the number of input nodes was determined with this training. The number of hidden number is calculated using equation 9 for different K values. The number of epochs is 100, learning rate and other parameters are same for each K principal component. The result is captured in Fig. 7, which shows the average cross validation accuracy versus the number of principal components.

H. Training Further with Best Principal Components (K)

After finding the number principal component with respect to the accuracy from second experiment, the three best K was chosen which gave the maximum average cross validation accuracy in cross validation. These models were further trained with different epochs and hidden nodes as shown in Table V.

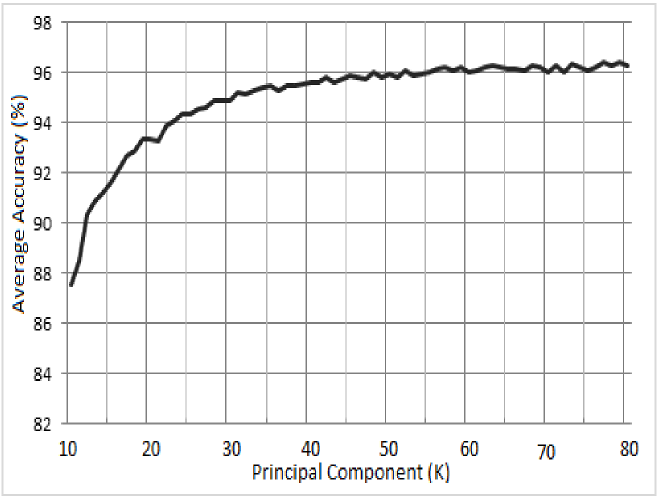


Fig. 7. Average cross validation accuracy vs number of principal component

TABLE V
THE RESULT OF NEURAL NETWORK WITH DIFFERENT K PRINCIPALS AND PARAMETERS

K	Epochs	Hidden nodes	Average cv Accuracy	Time Taken
52	500	31	96.51	5274.591
52	1000	31	96.95	9962.74
52	1000	70	97.97	24713.928
52	1000	90	98.02	27397.288
66	500	38	96.96	6250.179
66	1000	38	97.15	13109.53
66	1000	85	98.18	25847.239
66	1000	90	97.93	26464.829
66	1000	99	98.27	27273.124
77	500	44	97.05	7411.033
77	1000	44	97.19	14345.86
77	1000	70	98.01	26064.834
77	1000	90	98.15	31576.247
77	1000	100	98.18	31576.247

From the result, the model with K = 66, 1000 epochs, 99 hidden nodes is selected for testing on the MNIST's test data. Comparing the time taken to train the model with or without PCA, it is seen that using PCA the computation time reduces by 39.429% getting almost identical accuracy. If different K vale value is chosen then this comparison may differ.

Using the model selected, it was trained using the 60,000 training data and tested with 10,000 test data as the result shown in result section. To get best model submitted on

Kaggle, we also tried training the model by increasing the hidden nodes to large number which are 200, 300, and 500 and tested it with 10,000 test samples where we obtained the accuracy 98.23%, 98.33% and 98.67% respectively. We were able to do all these long experiments as we had powerful server to run all our trainings.

VI. Result

After cross validation, the algorithms with and without PCA with best accuracy were selected for testing and comparison. These models were trained using the 60,000 MNIST training sample and then tested using the 10,000 test samples. The algorithms are:

A. Neural Network Without PCA

The number of input nodes is 784 as there are 784 features, 397 in the hidden layer and 10 output nodes in output layer. The learning rate is 0.01 and regularization lambda is 2. The time taken to train the model is 27109.652seconds.

B. Neural Network With PCA

This is the proposed algorithm for this paper. The minimum principal component which gave the maximum accuracy was selected, that is 66 principal comments which retain about 92% of variance (see Fig. 6). Therefore, the number of input nodes is 66 based on principal component, 99 nodes in the hidden layer and 10 output nodes in output layer. The learning rate is 0.01 and regurgitation lambda is 2. The time taken to train the model is 4223.010seconds. The results of these 2 algorithms are shown in Table VI.

TABLE VI
THE COMPARISON OF NEURAL NETWORK RESULTS WITH AND WITHOUT PCA

Digit	**NN without PCA (784-397-10)**	**PCA + ANN**
0	99.39%	99.29%
1	99.12%	99.21%
2	98.16%	98.16%
3	98.51%	98.71%
4	98.17%	97.96%
5	97.09%	97.98%
6	98.43%	98.64%
7	98.25%	98.15%
8	97.95%	98.15%
9	97.82%	97.62%
Total	98.29%	98.39%

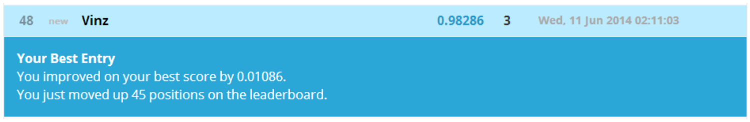
It is seen that using PCA with neural network, the computation is reduced by about 80% in this case and it would reduce further or less depending on the number of principal component selected. The proposed model has better accuracy rate compared to model without PCA.

C. Testing Using Paint Digit Image

The proposed algorithm was then used to predict 30 paint digit samples. The algorithms were able to successfully classify 96.7% of the test samples.

D. Testing in Kaggle Digit Recognition Completion

The proposed algorithm is also tested using the kaggle's testing dataset. The dataset consist of 28,000 testing images with 784 features (28 x 28 pixels). This is independent images whose labels are not given. Using the proposed algorithm, we classified each of the 28,000 samples and the predicted label/output for each sample was placed in the csv file with each row corresponding to each sample (28,000 rows). The csv file was uploaded on Kaggle and instant accuracy was given and it got ranked with other competitors in the leaderboard. Our proposed algorithm got 98.286% accuracy and is ranked 48 out of 384 competitors as shown in Fig. 8.


48 new Vinz 0.98286 3 Wed, 11 Jun 2014 02:11:03

Your Best Entry
You improved on your best score by 0.01086.
You just moved up 45 positions on the leaderboard.

Fig. 8. Ranking obtained from Kaggle Digit Recognition competition

VII. Discussion

The proposed system had 98.4% accuracy, which means 1.6% is misclassified image by the proposed algorithm which is shown in Fig. 9. On the list, some reasons for misclassification were made due to unclear way of writing, segmentation problem or noise in the data. To increase recognition further research is needed on ambiguous digits. There are some digits which are not misclassified but can be easily recognized by human, which means that the algorithm can be improved. Some improvement might be by adding new features, use feature extractions technique to remove noise make it clear for algorithm to classify.

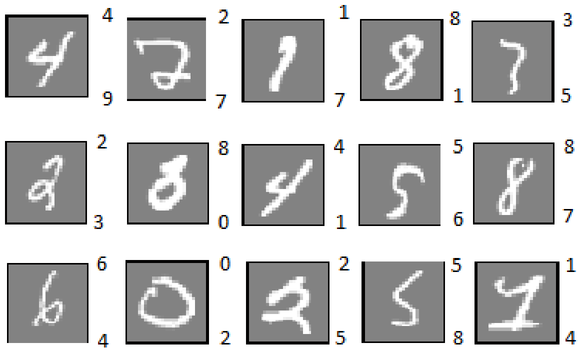


Fig. 9 Misclassified Digits. The top labeled digit is actual digit and the bottom labeled digit is predicted digit.

The proposed system has many computational advantages as it doesn't require must storage and done the training in 5 times faster than using neural network on its own. This means that the system can be trained using less powerful machine. The selection of number of principal component to be used in the model is a critical decision as this will also reflect on the accuracy based on the variance retained. We will expand this system in future to use genetic algorithm to reduce feature obtaining the best feature required to obtain the maximum accuracy.

The following table VII compared the proposed algorithm's accuracy with other algorithms used for digit recognition using Neural Networks.

TABLE VII
THE COMPARISON OF OTHER NEURAL NETWORK RESULTS WITH PROPOSED ALGORITHM

Algorithm	Accuracy
ANN with PCA (K=64, h= 35) [1]	91.20%
Boosted LeNet 4 [9]	99.3%
ANN with PCA (Proposed)	98.39%

VIII. Conclusion

In this paper, a method to recognize handwritten digit by using single layer neural network and PCA was proposed. The aim was to get maximum accuracy on digit recognition and to reduce the features for ease computation. This was achieved by the algorithm with 66 principal components giving 98.39% accuracy on MNIST test dataset.

Some misclassified digits are ambiguous either by unclear writing or segmentation problem. This can be further improved by increasing the feature and to use multiply feature extraction techniques. A combination of classifiers can also be used but we want the computation to be minimized.

More features extraction methods can be used to select the best feature required for digit recognition. In future, we will extend this system to use Genetic Algorithm to select the best features to obtained maximum accuracy in handwritten digit recognition, a computer vision problem.

References

[1] Z. Dan, and C. Xu, "The Recognition of Handwritten Digit Basedon BP Neural Network and the Impleementation on Android," Third International Conference on Intelligent System Design and Engineering Applications, pp. 1498-1501, 2013.

[2] B. J. Zwaag, "Handwritten Digit Recognition: A Neural Network Demo" B. Reusch (Ed.): Fuzzy Days2001, pp. 762–771, 2001.

[3] Cruz, G. Cavalcanti, and T. Ren, "Handwritten Digit Recognition Using Multiple Feature Extraction Techniques and Classifier Ensemble," 17th International Conference on Systems, Signals and Image Processing, pp. 215-218, 2010.

[4] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradientbased learning applied to document recognition," Proceedings of the IEEE, vol. 86, no. 11, pp. 2278–2324, 1998.

[5] P. Zhang, Reliable recognition of handwritten digits using a cascade ensemble classifier system and hybrid features, Ph.D. thesis, Concordia University, Montreal, P.Q., Canada, 2006.

[6] F. Lauer, C. Suen, and G. Bloch, "A trainable feature extractor for handwritten digit recognition," Pattern Recognition, vol. 40, no. 6, pp.1816–1824, 2007.

[7] E. Kavallieratou, K. Sgarbas, N. Fakotakis, and G. Kokkinakis, "Handwritten word recognition based on structural characteristics and lexical support," International Conference on Document Analysis and Recognition, pp. 562–567, 2003.

[8] L. S. Oliveira, R. Sabourin, F. Bortolozzi, and C. Suen, "Automatic recognition of handwritten numerical strings: A recognition and verification strategy," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 24, no. 11, pp. 1438–1454, 2002.

[9] Y. LeCun, L. Jackel, L. Bottou, A. Brunot, C. Cortes, J. Denker, H. Drucker, I. Guyon, U. Muller, and E. Sackinger, "Comparison of learning algorithms for handwritten digit recognition," International conference on artificial neural networks volume 60, 1995.

[10] B. Karlik and A. Olgac, "Performance Analysis of Various Activation Functions in Generalized MLP Architectures of Neural Networks," International Journal of Artificial Intelligence And Expert Systems (IJAE), Volume (1): Issue (4).

[11] Rani, D.K.U.: 'Analysis of Heart Diseases Dataset Using Neural Network Approach', International Journal of Data Mining & Knowledge Management Process (IJDKP), 2011, 1, (5)

[12] S. Knerr, L. Personnaz, G. Dreyfus, "Handwritten Digit Recognition by Neural Networks with Single-Layer Training," IEEE Transactions on Neural Networks, vol. 3, 962 (1992).

[13] A. Ilin, and T.Raiko,"Practical Approaches to Principal Component Analysis in the Presence of Missing Values" Journal of Machine Learning Research 11 (2010) 1957-2000.